Transferring visuomotor learning from simulation to the real world for manipulation tasks in a humanoid robot

Nguyen, Phuong D. H. ; Fischer, Tobias; Chang, Hyung Jin; Pattacini, Ugo; Metta, Giorgio; Demiris, Yiannis

DOI:
10.1109/IROS.2018.8594519

Transferring Visuomotor Learning from Simulation to the Real World for Robotics Manipulation Tasks

Phuong D.H. Nguyen[1,2], Tobias Fischer[2], Hyung Jin Chang[2,3], Ugo Pattacini[1], Giorgio Metta[1], Yiannis Demiris[2]

***Abstract*— Hand-eye coordination is a requirement for many manipulation tasks including grasping and reaching. However, accurate hand-eye coordination has shown to be especially difficult to achieve in complex robots like the iCub humanoid. In this work, we solve the hand-eye coordination task using a visuomotor deep neural network predictor that estimates the arm's joint configuration given a stereo image pair of the arm and the underlying head configuration. As there are various unavoidable sources of sensing error on the physical robot, we train the predictor on images obtained from simulation. The images from simulation were modified to look realistic using an image-to-image translation approach. In various experiments, we first show that the visuomotor predictor provides accurate joint estimates of the iCub's hand in simulation. We then show that the predictor can be used to obtain the systematic error of the robot's joint measurements on the physical iCub robot. We demonstrate that a calibrator can be designed to automatically compensate this error. Finally, we validate that this enables accurate reaching of objects while circumventing manual fine-calibration of the robot.**

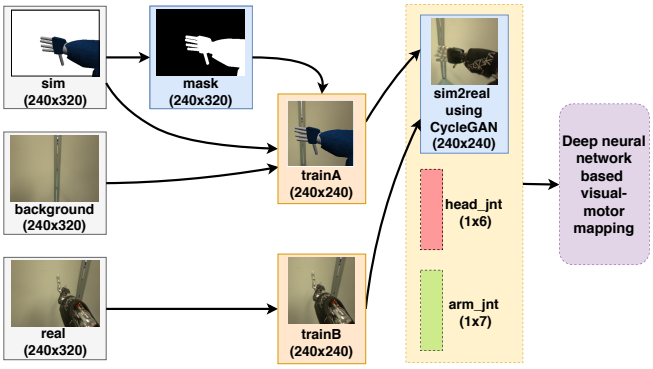

Fig. 1. Overview of the overall learning framework. Images obtained using a simulator are first being implanted with real background, and then CycleGAN [6] is used to synthesize realistically looking "sim2real" images. These are used as inputs to a deep neural network along with the head joints obtained from the simulator. The aim of the deep network is to estimate the arm joint configuration.

I. Introduction

Humans are able to achieve remarkably accurate and fast hand-eye coordination when performing tasks such as reaching and grasping. A key requirement for this skill is the association of a visual percept with proprioception by means of a visuomotor mapping [1]. It has been suggested that infants develop this visuomotor mapping during the development of reaching skills [2].

Most previous works in robotics find the visuomotor mapping in two steps [3]. The first step is to obtain the robot's kinematic model, which allows finding the position of a joint in Cartesian space given the joint configuration (*i.e.* the joint angles). If a physical model of the robot is available (*e.g.* a CAD model) the kinematic model can be represented in the Denavit–Hartenberg notation. An alternative is to learn the kinematic model without using *a priori* information, as exemplified in [4], [5].

The second step, called *visual-based pose estimation*, serves to improve the robustness and the accuracy of the first step and consists in finding the pose of the end-effector within the images acquired from the robot cameras. This is usually performed given a single image or a stereo pair of images [3]. In the stereo vision case, visual features are extracted to find the position of the end-effector in 2D space, and the disparity between the left and right images is used to find the 3D position with respect to the reference frame of the camera. Using the kinematic model obtained in the first step, this position can then be expressed in the robot's root reference frame.

However, the two-steps approach outlined above has several drawbacks. As highlighted in [7], there are various error sources involved in both steps. Within the first step, even if an accurate kinematic model exists, precise position control is hard to achieve using a cable driven robot like the iCub. Furthermore, some mechanical pieces such as the iCub eyes need to be re-calibrated at each startup of the robot. Similarly, the visual-based pose estimation step is often inaccurate as the end-effector appearance in an image is highly dependent on the head pose that itself underlies uncertainties for the same reasons as outlined above. The visual-based pose estimation step also relies on precise estimates of the camera parameters, which is tackled elsewhere [8]. To minimize the effect of these inaccuracies, an additional calibration step has been suggested to find a set of offsets specifically for hand-eye coordination [3], [8]–[10].

We propose to find the visuomotor mapping in a single step rather than considering the two problems independently and finding an offset mapping subsequently as outlined

[1]Phuong D.H. Nguyen, Ugo Pattacini, and Giorgio Metta are with the iCub Facility, Istituto Italiano di Tecnologia, Genova 16163, Italy {phuong.nguyen, ugo.pattacini, giorgio.metta}@iit.it

[2]Phuong D.H. Nguyen, Tobias Fischer, Hyung Jin Chang, and Yiannis Demiris are with the Personal Robotics Lab, Imperial College, London SW7 2AZ, UK {t.fischer, hj.chang, y.demiris}@imperial.ac.uk

[3]Hyung Jin Chang is also with the Intelligent Robotics Lab, School of Computer Science, University of Birmingham, Birmingham B15 2TT, UK

above. More specifically and as shown in Figure 1, we suggest to learn the mapping from an imprecise model in simulation using two components: 1) A deep neural network estimates the arm's joint configuration given images captured with the two eyes of the simulated robot and the corresponding head configuration. 2) To allow application of the deep neural network in the real world, the domain gap needs to be bridged as the image statistics between simulation and real world differ significantly. We propose to use an image-to-image translation method to bridge the domain gap.

The efficiency of our proposed visuomotor mapping method is exemplified on the robot calibration task. Based on the visuomotor mapping, we build a compensation model using the difference between the predicted joint values and the measured values of the joint encoders. We show that this ensures coordinated control of eye movement with hand movement on the physical robot. Our work could thus be used to sidestep the manual calibration that is needed for many manipulation tasks such as reaching as grasping.

II. Related Works

Learning a visuomotor mapping: Some methods closely follow the classical approach for visuomotor mapping outlined in the introduction, but use machine learning methods to make the mapping more robust and adaptable. For example, [11] relies on several radial basis functions. However, the method is limited in the sense that markers are required for feature extraction and the disparity is assumed to be known.

To find the end-effector pose using visual information, it has been suggested to compare the hand perceived by the robot's camera with a realistically rendered hand from simulation [12], [13]. A particle filter is then used to predict the 6D pose of the robot's hand, which is used for a visual servoing reaching task.

The hand-eye coordination task has also been investigated within the developmental robotics domain. Aguilar and Perez [14] propose a method which allows for coordination of the visual and tactile modalities, *i.e.* the robot's hand is equipped with tactile sensors which can perceive the presence of objects using touch. The focus of this method is on the emergence of higher-level behaviors such as the exploration of various objects and following objects using attentional processes. Similarly, Hwang *et al.* [15] investigate the emergence of a mirror neuron system for imitation learning using such a developmental approach. While [15] is limited to imitation learning using the same robot, Chang *et al.* [16], [17] present a method that allows to find body shape correspondence between a number of robots and humans in images. However, this method relies on motion information and is not suited to finding matches of images where the bodies are in the same pose.

Simulation to real transfer learning: One of our aims is to transfer skills from simulation to real robots, and several previous works have shown that this approach is viable. Rusu *et al.* [18] focus on a reaching task using reinforcement learning. They use a concept called progressive networks that allow to train parts of the network in simulation and then using these parts to bootstrap another network that is applied on the physical robot. Interestingly, the method learns directly from raw visual input, and is capable of using additional input modalities of the real robot that are not present in the simulator. Tzeng *et al.* [19] propose to use a domain confusion loss for end-effector pose estimation. The idea is to enforce the same feature representation regardless whether the input is from simulation or the real robot. This is implemented using a discriminator that attempts to classify whether the feature originated from a simulated or real image.

Several works tackle the problem that the control policy learned on simulation data does not directly map to the real robot. Christiano *et al.* [20] use a deep inverse model trained on the behavior of the real robot to find the action that resembles the high-level properties of the policy found in simulation, while abstracting away the low-level properties that differ between the simulation and the real environment. This approach was extended by Sadeghi *et al.* [21] who focus on finding a viewpoint-invariant control policy. Zhang *et al.* [22] propose a modular deep network for robot reaching. A perception module estimates the target object location, which is then used by a control module that issues velocity commands to reach that target. They improve hand-eye coordination accuracy by end-to-end fine-tuning of these modules.

A completely different approach is taken by Tobin *et al.* [23] and James *et al.* [24] who propose that generalization abilities can be achieved by alterations of the rendered images of the simulator in terms of textures, lighting conditions, camera position and other factors. A distinct property of these methods is that they do not require any fine-tuning on real images. This is in stark contrast to [25] where it is proposed to map the real world images back to the simulator domain rather than the more common method of finding a mapping into the real domain.

The work most similar to ours is that of Bousmalis *et al.* [26]. As in our method, a generator is trained to produce realistic images from synthetic images, and a discriminator is trained to distinguish synthetically created images from real images. However, their proposal is constrained to the grasping task as an estimate of the grasping success is used as semantic input when training the generator. As only the grasping success is considered, this relaxes the requirement of accurately estimating the state of each joint. Our method is complementary to these task specific methods, as our method aims to sidestep the manual calibration that is required in complex robots like the iCub, while being agnostic to the specific manipulation task.

III. Methodology

In this section, we first present the *Imperial-sim2real* dataset that contains both images obtained in simulation and from the physical robot (Section III-A). We then introduce an image-to-image translation method that maps images from the simulation domain to the real domain (Section III-B).

Finally, we show that these realistically looking images can be used to learn the visuomotor mapping (Section III-C). All code is available on our GitHub repository[1].

A. Action-based dataset generation

To create the *Imperial-sim2real* dataset, we use the iCub humanoid robot [27]. The iCub has the size of a three to four year old child and was designed to study embodied cognition and autonomous exploration. In total, the iCub has 53 Degrees of Freedom (DoF), and in the scope of this paper we focus on the 7 DoFs of each arm and the 6 DoFs of the iCub's head. Both eyes are equipped with a RGB camera that allows for stereo vision capabilities [8].

Dataset overview: The *Imperial-sim2real* dataset (doi: 10.5281/zenodo.1186943) contains the following elements:

1) *Sim*: Stereo vision image pairs of the robot's arm in simulation, along with the corresponding head joint and arm joint configurations.
2) *Background*: Background images collected using the physical robot without the robot arm in the visual field of view, with corresponding head joint configuration.
3) *TrainA*: Images created by combining *Sim* images and *Background* images of similar head configurations. Here the *Background* image with most similar head configuration to the *Sim* image is used.
4) *TrainB*: Stereo vision image pairs of the physical robot's arm.
5) *sim2real*: Synthetic images that were translated from the simulation domain into the real domain using an image-to-image translation method. Specifically, *TrainA* images are translated using *CycleGAN* as described in Section III-B.

Motor babbling for randomized actions: As we want to find the mapping of the visual and motor spaces in one shot, we aim to acquire data from the corresponding sensor sources (stereo vision image pairs, head configuration and arm configuration) in the whole working space of the robot's arm using a motor babbling scheme.

More precisely, we cover the working space by issuing random actions to the relevant joints of the kinematic chain (*i.e.* from the eyes to the robot's hand) and store the stereo image pair as well as the measurements of the joint encoders (7 arm joints and 6 head joints). For simplicity, we only find the visuomotor mapping for the right hand and ensure that the left hand is out of the field of view (the mapping for the left hand can be found in the same way). We generate the arm motions using a velocity controller following the proposal of Zambelli and Demiris [4], as shown in Eq. (1), and extend their method to also move the head joints using a position controller, as shown in Eq. (3). We utilize the superscripts a and h for arm and head respectively.

Arm velocity controller: The velocity command $\mathbf{v}^a$ is found as follows:

$$\mathbf{v}^a = K \cdot \left(\mathbf{p}^a_{\text{ref}} - \mathbf{p}^a_t\right), \tag{1}$$

[1] https://github.com/robotology/visuomotor-learning

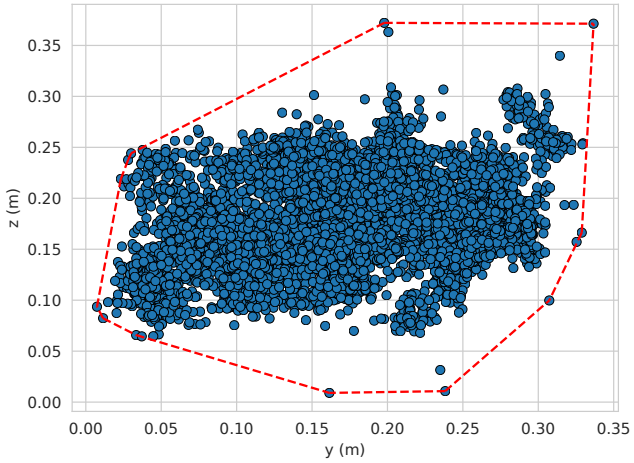

Fig. 2. Approximation of the right arm coverage achieved by issuing random motor commands in simulation. The z-axis points upward, and the y-axis points left to right. The motion in the x-axis (pointing towards the robot) is negligible with regards to the other two directions and is not shown for simplicity. The dashed red line shows the coverage of the babbling algorithm, while the blue dots represent individual data points of the end-effector position.

with $\mathbf{p}^a_t$ being a vector of values containing the measurements of arm joints at time t, K is the proportional gain, and the reference values $\mathbf{p}^a_{\text{ref}}$ of the arm joints are generated by:

$$\mathbf{p}^a_{\text{ref}} = \mathbf{p}^a_0 + \widetilde{A} \cdot \sin\left(2\pi \widetilde{f} t\right) \cdot \mathbb{1}, \tag{2}$$

where $\mathbb{1}$ duplicates a scalar into a vector of the appropriate size. The initial guess $\mathbf{p}^a_0$ ensures that the right arm is visible in the whole motion sequence, and $\widetilde{A} \sim \mathcal{N}(0, A)$ and $\widetilde{f} \sim \mathcal{N}(0, f)$ are normally distributed parameters for the amplitude and frequency respectively.

Head position controller: The head position $\mathbf{p}^h$ is determined according to

$$\mathbf{p}^h = \mathbf{p}^h_0 + \widetilde{H} \cdot \mathbb{1}, \tag{3}$$

where $\mathbf{p}^h_0$ denotes the random initial head configuration and $\widetilde{H} \sim \mathcal{N}(0, H)$ is the normally distributed gain of the position controller. For safety reasons, the final control value sent to each joint is constrained by the firmware's bounding value.

Workspace coverage: Covering a sufficiently large working space requires setting large values for A, H and f. While this is feasible in simulation, lower values have to be used on the real robot due to mechanic stress constraints. For this work, we set the parameters as following: $A = 5$, $f = 0.2$ and $H = 10$ in simulation or $H = 5$ for the physical robot. Thus using the iCub simulator [28] has the additional benefit that it allows the collection of a larger, more diverse dataset compared to the physical robot. The resulting arm coverage in the simulated workspace is shown in Figure 2, with the arm's sweeping volume approximated as $V \approx S_{convex} \cdot \bar{d} = 0.39m^3$ (where S_{convex} is the area covered by the convex hull spanned in the plane of the y- and z-axes, and $\bar{d}$ is the average distance between the robot's end-effector and the robot's shoulder in direction of the x-axis [motion in the x-axis is negligible]).

B. Image-to-image translation from the simulator to the real domain

In this section we learn a mapping function that maps images containing the iCub's arm from the simulation domain to the real domain. In the computer vision domain, image-to-image translation has been addressed as learning a mapping function between an input image $\{\mathbf{a}_i\}_{i=1}^{N} \in TrainA$ and an output image $\{\mathbf{b}_i\}_{i=1}^{N} \in TrainB$ by training aligned image pairs [29]. In the following, we will abbreviate $TrainA$ with $\mathbf{A}$ and $TrainB$ with $\mathbf{B}$ for brevity. However, in our task we only have two independent sets of images, where one consists of simulation arm images $\mathbf{A}$ and the other consists of real robot arm images $\mathbf{B}$ – there is no paired data indicating how a simulated image could be translated to a corresponding real image. In fact, as outlined in the introduction, we cannot rely on the joint configuration obtained by the physical robot as there are various unavoidable sources of errors.

Recently, generative adversarial network (GAN) [30] based methods have shown good performance in generating realistic images, and some variants utilizing GANs have been applied to the image translation task without requiring aligned image pairs [6]. In this work, we adopt a state-of-the-art image-to-image translation method called Cycle-Consistent Adversarial Network (*CycleGAN*) [6]. The *CycleGAN* is based on a combination of adversarial loss $\mathcal{L}_{\text{GAN}}$ and cycle consistency loss $\mathcal{L}_{\text{cyc}}$ in order to learn two mapping functions $G_{\mathbf{B}}\colon \mathbf{A} \to \mathbf{B}$ and $G_{\mathbf{A}}\colon \mathbf{B} \to \mathbf{A}$. In the following, we briefly describe the architecture and implementation of *CycleGAN* in our work.

Firstly, two adversarial discriminative networks $D_{\mathbf{A}}$ and $D_{\mathbf{B}}$ are introduced. $D_{\mathbf{A}}$ distinguishes between real images $\{\mathbf{a}\}$ and translated images $\{G_{\mathbf{A}}(\mathbf{b})\}$ and similarly $D_{\mathbf{B}}$ distinguishes between images $\{\mathbf{b}\}$ and $\{G_{\mathbf{B}}(\mathbf{a})\}$. The generators $G_{\mathbf{A}}$ and $G_{\mathbf{B}}$ try to generate images $G_{\mathbf{A}}(\mathbf{b})$ and $G_{\mathbf{B}}(\mathbf{a})$ that look similar to images from the $\mathbf{A}$ and $\mathbf{B}$ domains respectively. The adversarial loss for the mapping $G_{\mathbf{B}}\colon \mathbf{A} \to \mathbf{B}$ is

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(G_{\mathbf{B}}, D_{\mathbf{B}}, \mathbf{A}, \mathbf{B}) = \; & \mathbb{E}_{\mathbf{a}\sim P_{\mathbf{a}}}\Big[\log\Big(1 - D_{\mathbf{B}}\big(G_{\mathbf{B}}(\mathbf{a})\big)\Big)\Big] \\ & + \mathbb{E}_{\mathbf{b}\sim P_{\mathbf{b}}}\Big[\log D_{\mathbf{B}}(\mathbf{b})\Big], \end{aligned} \tag{4}$$

and the mapping $G_{\mathbf{A}}\colon \mathbf{B} \to \mathbf{A}$ is similarly designed as $\mathcal{L}_{\text{GAN}}(G_{\mathbf{A}}, D_{\mathbf{A}}, \mathbf{B}, \mathbf{A})$.

Secondly, the cycle consistency loss is designed to regularize the mappings, *i.e.* translation from one domain to the other, by ensuring that translating an already translated image back to the original domain should resemble the original image: $\mathbf{a} \to G_{\mathbf{B}}(\mathbf{a}) \to G_{\mathbf{A}}\big(G_{\mathbf{B}}(\mathbf{a})\big) \approx \mathbf{a}$. The cycle consistency loss is formulated as

$$\begin{aligned}\mathcal{L}_{cyc}(G_{\mathbf{B}}, G_{\mathbf{A}}) = \; & \mathbb{E}_{\mathbf{a}\sim P_{\mathbf{a}}}\Big[\big\|G_{\mathbf{A}}\big(G_{\mathbf{B}}(\mathbf{a})\big) - \mathbf{a}\big\|_1\Big] \\ & + \mathbb{E}_{\mathbf{b}\sim P_{\mathbf{b}}}\Big[\big\|G_{\mathbf{B}}\big(G_{\mathbf{A}}(\mathbf{b})\big) - \mathbf{b}\big\|_1\Big], \end{aligned} \tag{5}$$

where $\|\cdot\|_1$ indicates the l_1-norm. By merging the adversarial losses and cycle consistency loss, the full loss becomes

$$\begin{aligned}\mathcal{L}(G_{\mathbf{A}}, G_{\mathbf{B}}, D_{\mathbf{A}}, D_{\mathbf{B}}) = \; & \mathcal{L}_{GAN}(G_{\mathbf{B}}, D_{\mathbf{B}}, \mathbf{A}, \mathbf{B}) \\ & + \mathcal{L}_{GAN}(G_{\mathbf{A}}, D_{\mathbf{A}}, \mathbf{B}, \mathbf{A}) \\ & + \lambda\, \mathcal{L}_{cyc}(G_{\mathbf{B}}, G_{\mathbf{A}}), \end{aligned} \tag{6}$$

where λ controls the relative weight of the adversarial losses and the cycle consistency loss.

We want to emphasize the importance of experimentally finding the correct value for λ in our setting. If λ is chosen too high, the *CycleGAN* tends to remember the learned images rather than generating new ones (in terms of interpolation and extrapolation). On the other hand, if λ is chosen too low, the arm pose is not closely resembled in the synthesized images. We experimentally found that $\lambda = 10$ represents a good compromise.

We implement the model in Tensorflow and train it with the Adam optimizer [31] using a learning rate of 0.0002 for both generators and discriminators.

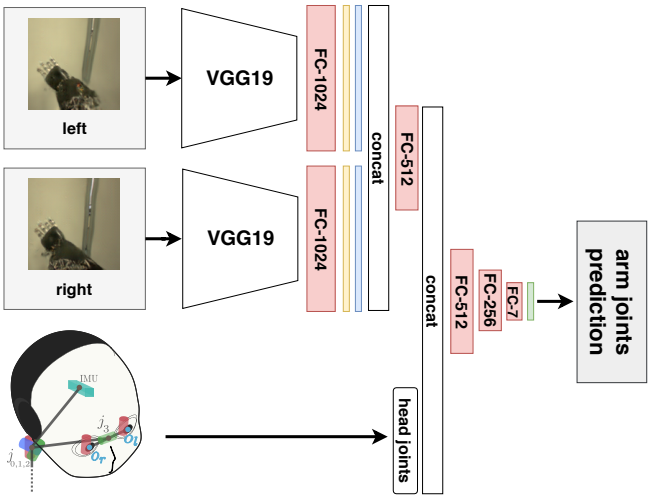

Fig. 3. Visuomotor deep neural network model. Yellow, blue and green layers denote *batch normalization*, *ReLU* and *tanh* respectively. We abbreviate fully-connected layers with "FC" and "concat" stands for concatenation.

C. Learning the visuomotor mapping

We treat the visuomotor learning as a regression problem, where given a pair of images containing the robot's arm (from the stereo-vision system) and the head joint configuration of the robot (neck-eyes), the aim is to estimate the corresponding joint configuration of the arm. Therefore, an input sample ($\mathbf{x} \in \mathbf{X}$) contains two images and six measurements of the head joints, while a corresponding output sample ($\mathbf{y} \in \mathbf{Y}$) contains seven measurements of the robot's arm joints. We assume that there is only one of the robot's two arms contained in the input samples. The regression problem is then to approximate the function f to map the input and output, *i.e.* $\mathbf{Y} = f(\mathbf{X})$.

To estimate the mapping function f, we propose a deep neural network model as depicted in Figure 3. It contains the following components:

- Two identical VGG19 [32] networks to extract features from images obtained by the left eye camera and right

eye camera respectively. The feature vector of each VGG19 network is first routed to a dense layer of 1024 fully connected units, followed by a *batch normalization* layer and a *ReLU* activation layer. The dense layers are then concatenated and integrated by a dense layer of 512 units. This deep visual output can be considered as implicit 3D estimate from the raw stereo image pair.

- The head joint configuration is concatenated to this deep visual output, and then further processed in a densely connected network composed of three layers with 512, 256 and 7 units respectively. Finally, the predicted joint values $\widehat{\mathbf{Y}}$ are obtained after applying a *tanh* activation layer.

Contrary to the classical approach, which maps a visually perceived object to Cartesian space, and then from Cartesian space to the motor space, our mapping f directly infers the configuration of the arm joints in motor space from the spatial relation of pixels in raw images. Thus our method does not suffer from the infinite solution problem of the inverse kinematics transformation (in redundancy manipulators) as in the latter step of the classical approach. In other words, our method considers a robot's arm in the visual space as a whole rather than as a single, representative point in Cartesian space (also called tool center point).

The training process consists of two steps. First, to bootstrap the network, we only use the *Sim* dataset which contains the unmodified simulator images. This results in the *sim* predictor. We then fine-tune this network further using samples from the *sim2real* dataset (obtained using *CycleGAN*) which results in the *real* predictor. We chose this training scheme as the *sim2real* dataset is significantly smaller compared to the *Sim* dataset (some images are discarded in the *sim2real* dataset as described in Section IV-B).

The deep neural network model is implemented in Keras with Tensorflow back-end, and is trained with the *mean squared error* loss function using the Adam optimizer [31]. We use a batch size of 160 and a learning rate of 0.0001.

IV. Experiments & Results

In this section, we evaluate our proposed method in a number of settings. The first experiment purely evaluates the deep neural network used for the visuomotor mapping (Section IV-A). The second experiment evaluates the image-to-image translation method and shows that realistic images can be generated from simulation while maintaining the semantic information of the arm pose (Section IV-B). The third experiment is a combination of the first two, and evaluates the visuomotor mapping in the real domain, *i.e.* using the physical robot and the *sim2real* dataset (Section IV-C). Finally, we show that a simple calibrator using the offsets between the measured joint values and predicted joint values can be used in an object reaching scenario (Section IV-D).

A. Robot hand visual tracking in simulation

In the first experiment, we evaluate the visuomotor deep neural network that was presented in Section III-C in simulation. The model is trained with $N = 34000$ training samples obtained by the proposed motor babbling scheme. The training input consist of a stereo image pair with the corresponding head configuration, and the network is trained to estimate the arm's joint values.

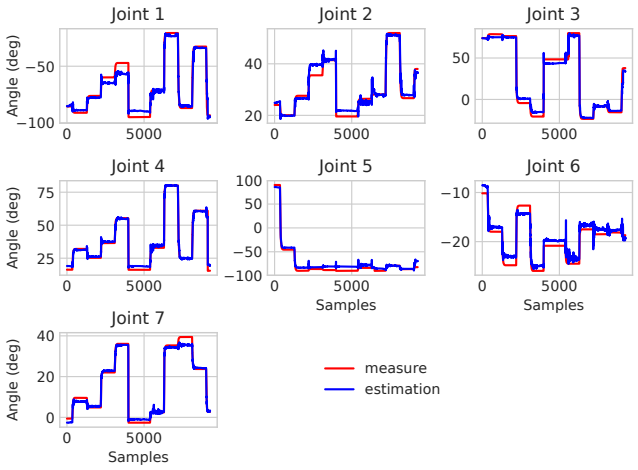

Fig. 4. Arm joint estimation from visual input in the iCub simulator. Each plot corresponds to one of the seven arm joints of the iCub and shows the measured joint value (red) and predicted joint value (blue) over time. As all joints are estimated accurately, this figure validates the efficiency of the proposed visual-motor deep neural network.

The trained model is then tested on a testing set separate from the training and validation sets (containing 6000 samples). If the model is able to generalize to new joint configurations, the error between the measured joints of the simulator and the predicted joints should be small. Indeed, Figure 4 confirms that this is the case. The average error is 1.85 ± 1.32 degrees.

This result also demonstrates that the proposed deep network can be used as a component within action planning without having to rely on inverse kinematics methods. For example, the network can be used to find the corresponding joint configuration of the robot's arm touching an object. Then, it is straightforward to issue motor commands to achieve this joint configuration.

Another advantage of the proposed visuomotor deep neural network is that the inference only takes $\approx 10ms$ (on a NVIDIA GTX 1080Ti), which is significantly faster than other approaches for hand-eye coordination on the iCub (*e.g.* $> 100ms$ in [12]).

B. Imperial-sim2real dataset

The synthetic images presented in this section are generated by the image-to-image translation method described in Section III-B. The *Sim* element contains approximately 40000 training samples collected using the simulator, and there are over 8000 images in the *Background* and *TrainA* elements of the dataset. The *TrainB* element, which is collected using the physical robot, contains more than 4000 stereo image pairs. Originally we collected 18000 stereo image pairs, however we found that using fewer pairs provides sufficient training data for the *CycleGAN*, reduces training time and avoids overfitting. The *sim2real* element contains approximately 9000 synthetic images. While theoretically we could obtain as many synthetic images as there are in the

Fig. 5. Image-to-image translation results. Left: simulator images where the background was replaced with random background images of the real domain (*TrainA* set). Middle: corresponding realistically looking images synthesized using CycleGAN (*sim2real* set). The pose of the synthesized arm closely resembles that of the simulator arm. Right: the real images that are most similar to the synthesized ones in the middle column (*TrainB* set).

Sim image set, we manually discard images which are too blurry or where the synthetic arm has an unrealistic pose. In Figure 5, we show some examples of images acquired using the iCub simulator (*TrainA* images), and their corresponding images in the real domain acquired using CycleGAN (*sim2real* images). The *Imperial-sim2real* dataset is made available to the public (doi: 10.5281/zenodo.1186943).

C. Robot hand visual tracking on the physical robot

In our third experiment, we investigate whether the tracking also performs well on the real robot. Compared to experiment in Section IV-A, the visuomotor deep network is trained with synthetic images obtained using CycleGAN as described in Section III-B (*sim2real* images). The network is then tested on the physical robot, and Figure 6 shows the error between the predicted and measured joint states. As expected, there are some discrepancies due to the imprecise calibration of the physical robot. Importantly, however, in the next section we show that the discrepancies are systematic, and we construct a calibrator to compensate for these discrepancies.

D. Joint calibration on the physical robot

While the papers main contribution is to learn the visuomotor mapping as described in Section III, here we investigate the accuracy of our mapping using a simple calibrator. The idea is as follows: the motor babbling scheme (Section III-A) is performed on the physical robot, and the images of the arm are recorded alongside the measured joint values $\mathbf{y}$. Using the collected images and the *real* predictor (Section III-C), we obtain (paired) predicted joint values $\widehat{\mathbf{y}}$. We then map the measured joint values to the predicted joint

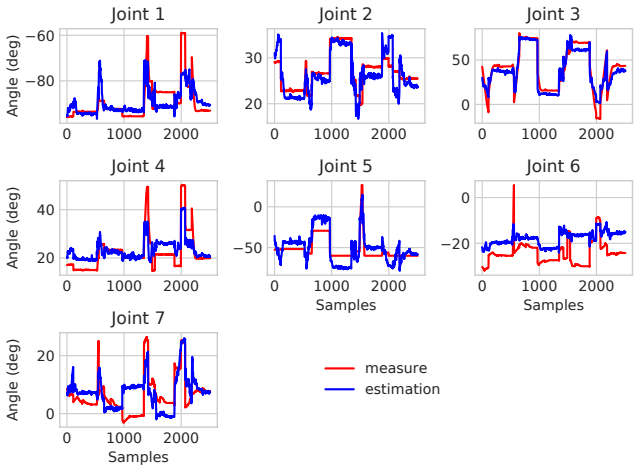

Fig. 6. Arm joint estimation from visual input on the physical robot using our proposed visuomotor deep neural network trained on samples produced by the CycleGAN. The small discrepancies are due to inaccurate calibration of the real robot, and can be compensated by a calibrator (see Section IV-D).

values using a simple polynomial. One example application shown in this section is object touching. Specifically, the polynomial can be used to map a target joint configuration $\mathbf{y}^*$ obtained from inverse kinematics to the corrected joint configuration $\widehat{\mathbf{y}}^*$ that is used to touch the object (rather than using $\mathbf{y}^*$ directly).

The calibrator is implemented as polynomial $\widehat{\mathbf{y}} = \mathbf{c}_0 + \mathbf{c}_1 \odot \mathbf{y} + \mathbf{c}_2 \odot \mathbf{y}^2 + \mathbf{c}_3 \odot \mathbf{y}^3$, where $\widehat{\mathbf{y}}$ denotes the predicted joint values, $\mathbf{y}$ denotes the measured joint values and $\odot$ denotes the Hadamard product[2]. The $\mathbf{c}_i$ are constant vectors that are approximated using the collected data in Section IV-C by applying the *least squares polynomial fitting* method. This procedure can be fully automated and has to be repeated every time the mechanical properties of the robot change, *e.g.* after a cable has been replaced.

This calibrator can then be used to accurately touch objects in a table-top scenario. Using the stereo-vision and the eyes-to-root transformation of the robot [8], we obtain the Cartesian coordinates *(x,y,z)* of a known object in the root reference frame. The Cartesian coordinates are then converted to the corresponding target joint configuration $\mathbf{y}^*$ using the inverse kinematic library of the robot (iKin, see [33]). We want to emphasize that the inverse kinematics library is only used to obtain the joint configuration corresponding to Cartesian coordinates, but not to control the robot. We use the calibrator to obtain the corrected joint configuration $\widehat{\mathbf{y}}^*$ that is then issued to the robot.

We evaluate the reaching accuracy by varying the target object's position and comparing the success with and without using the proposed calibration method. Figure 7 shows that the robot fails to touch the object when directly using $\mathbf{y}^*$ (*i.e.* without usage of the proposed calibrator). When the corrected joint values $\widehat{\mathbf{y}}^*$ are used, the robot can successfully touch the object. The supplementary video provides further details on this validation[3].

[2]We abuse notation and denote $\mathbf{y} \odot \mathbf{y}$ as $\mathbf{y}^2$ and $\mathbf{y} \odot \mathbf{y} \odot \mathbf{y}$ as $\mathbf{y}^3$.
[3]`https://youtu.be/VMw8sVztcKA`

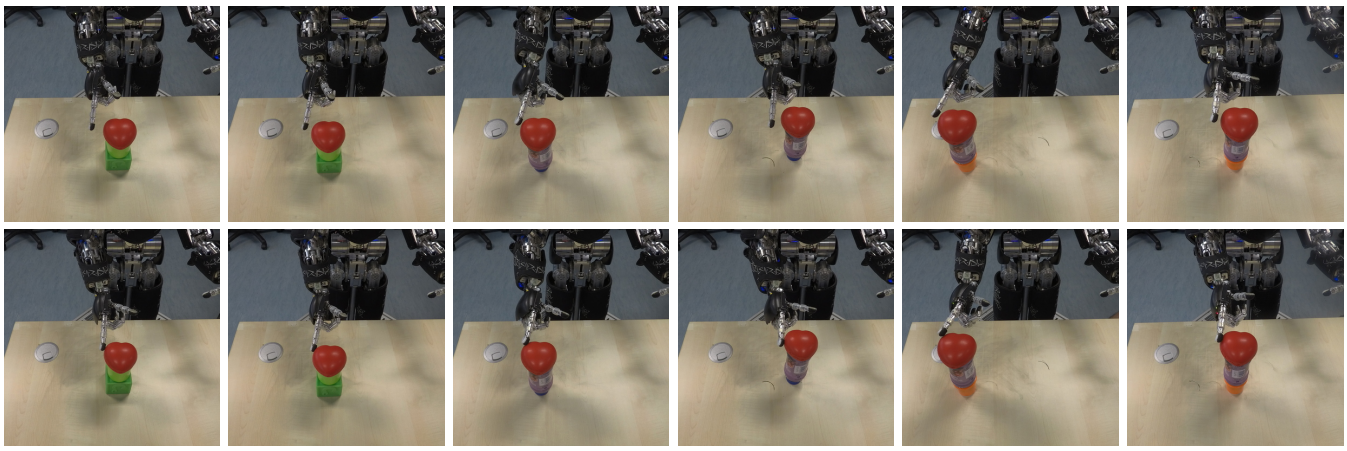

Fig. 7. Touching trial with and without calibration in different target position: First row–no calibration; Second row–with calibration

V. Conclusions

In this paper, we have introduced a new way of using data obtained using a simulator in the real domain. We have shown that synthesized data can be used to predict the robot's arm configuration given a pair of stereo images and the head configuration. The discrepancies between these predictions and the measured arm joint values can be used to train a calibrator, which is then used in a reaching task.

Our next step will be applying the learned visuomotor mapping in vision-based action planning, including reaching with obstacle avoidance and object grasping. For the reaching with avoidance task, we plan to use the proposed method to produce imagined images of the robot's arm, which can be checked against collisions with objects. An advantage of our method is that the planning problem is mapped from vision space to joint space in real-time, where well-established motion planning techniques such as Rapidly exploring Random Trees (RRT*) and Probabilistic RoadMap (PRM*) [34] can be applied. Another benefit of our framework is the feasibility of integrating the touch modality using the tactile sensors of the iCub.

One limitation of our method is that it is currently constrained to setups where the background closely resembles that of the dataset. We will overcome this limitation using domain randomization techniques [23], [24]. Other future works include the simultaneous learning of the visuomotor mapping for both arms, extending the framework to learn directly from uncalibrated images, and simultaneously learning the joint state and end-effector position. To achieve these goals we will be using methods from the multi-task learning [35] and transfer learning [36] fields.

Acknowledgments

Phuong D.H. Nguyen was supported by a Marie Curie Early Stage Researcher Fellowship (H2020-MSCA-ITA, SECURE 642667). We would like to thank Alessandro Roncone for the initial figure of the simplified iCub's head-eyes joints.

References

[1] R. S. Johansson, G. Westling, A. Bäckström, and J. R. Flanagan, "Eye–hand coordination in object manipulation," *Journal of Neuroscience*, vol. 21, no. 17, pp. 6917–6932, 2001.

[2] D. Corbetta, S. L. Thurman, R. F. Wiener, Y. Guan, and J. L. Williams, "Mapping the feel of the arm with the sight of the object: On the embodied origins of infant reaching," *Frontiers in Psychology*, vol. 5, no. 576, pp. 1–19, 2014.

[3] J. Hollerbach, W. Khalil, and M. Gautier, "Model Identification," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds., 2008, pp. 321–344.

[4] M. Zambelli and Y. Demiris, "Online Multimodal Ensemble Learning using Self-learnt Sensorimotor Representations," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 9, no. 2, pp. 113–126, 2017.

[5] M. Antonelli, E. Chinellato, and A. P. Del Pobil, "Implicit mapping of the peripersonal space of a humanoid robot," in *IEEE Symposium on Computational Intelligence, Cognitive Algorithms, Mind, and Brain*, 2011, pp. 143–150.

[6] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks," in *IEEE International Conference on Computer Vision*, 2017, pp. 2223–2232.

[7] J. Kim, K. Iwamoto, J. J. Kuffner, Y. Ota, and N. S. Pollard, "Physically based grasp quality evaluation under pose uncertainty," *IEEE Transactions on Robotics*, vol. 29, no. 6, pp. 1424–1439, Dec. 2013.

[8] S. R. Fanello, U. Pattacini, I. Gori, *et al.*, "3D stereo estimation and fully automated learning of eye-hand coordination in humanoid robots," in *IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 1028–1035.

[9] R. Horaud and F. Dornaika, "Hand-Eye Calibration," *International Journal of Robotics Research*, vol. 14, no. 3, pp. 195–210, 1995.

[10] G. D. Hager, W.-C. Chang, and A. S. Morse, "Robot hand-eye coordination based on stereo vision," *IEEE Control Systems*, vol. 15, no. 1, pp. 30–39, 1995.

[11] M. Antonelli, E. Chinellato, and A. P. Del Pobil, "Online learning of the visuomotor transformations on a humanoid robot," in *Intelligent Autonomous Systems*, vol. 12, 2013, pp. 853–861.

[12] P. Vicente, L. Jamone, and A. Bernardino, "Online Body Schema Adaptation Based on Internal Mental Simulation and Multisensory Feedback," *Frontiers in Robotics and AI*, vol. 3, no. 7, pp. 1–20, 2016.

[13] C. Fantacci, U. Pattacini, V. Tikhanoff, and L. Natale, "Visual end-effector tracking using a 3d model-aided particle filter for humanoid robot platforms," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 1411–1418.

[14] W. Aguilar and R. P. y. Pérez, "Emergence of eye–hand coordination as a creative process in an artificial developmental agent," *Adaptive Behavior*, vol. 25, no. 6, pp. 289–314, 2017.

[15] J. Hwang, J. Kim, A. Ahmadi, M. Choi, and J. Tani, "Predictive Coding-based Deep Dynamic Neural Network for Visuomotor Learning," in *Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics*, 2017, pp. 132–139.

[16] H. J. Chang, T. Fischer, M. Petit, M. Zambelli, and Y. Demiris, "Kinematic Structure Correspondences via Hypergraph Matching," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4216–4225.

[17] H. J. Chang, T. Fischer, M. Petit, M. Zambelli, and Y. Demiris, "Learning Kinematic Structure Correspondences Using Multi-Order Similarities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, to appear, 2017.

[18] A. A. Rusu, M. Večerík, T. Rothörl, *et al.*, "Sim-to-Real Robot Learning from Pixels with Progressive Nets," in *Conference on Robot Learning*, 2017, pp. 262–270.

[19] E. Tzeng, C. Devin, J. Hoffman, *et al.*, "Towards adapting deep visuomotor representations from simulated to real environments," *arXiv preprint arXiv:1511.07111*, 2015.

[20] P. Christiano, Z. Shah, I. Mordatch, *et al.*, "Transfer from Simulation to Real World through Learning Deep Inverse Dynamics Model," *arXiv preprint arXiv:1610.03518*, 2016.

[21] F. Sadeghi, A. Toshev, E. Jang, and S. Levine, "Sim2Real View Invariant Visual Servoing by Recurrent Control," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4691–4699.

[22] F. Zhang, J. Leitner, M. Milford, and P. Corke, "Sim-to-real transfer of visuo-motor policies for reaching in clutter: Domain randomization and adaptation with modular networks," *arXiv preprint arXiv:1709.05746*, 2017.

[23] J. Tobin, R. Fong, A. Ray, *et al.*, "Domain randomization for transferring deep neural networks from simulation to the real world," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 23–30.

[24] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," in *Annual Conference on Robot Learning*, 2017, pp. 334–343.

[25] J. Zhang, L. Tai, Y. Xiong, *et al.*, "VR Goggles for Robots: Real-to-sim Domain Adaptation for Visual Control," *arXiv preprint arXiv:1802.00265*, 2018.

[26] K. Bousmalis, A. Irpan, P. Wohlhart, *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 4243–4250.

[27] G. Metta, L. Natale, F. Nori, *et al.*, "The iCub humanoid robot: An open-systems platform for research in cognitive development," *Neural Networks*, vol. 23, no. 8-9, pp. 1125–1134, 2010.

[28] V. Tikhanoff, A. Cangelosi, P. Fitzpatrick, *et al.*, "An open-source simulator for cognitive robotics research: The prototype of the iCub humanoid robot simulator," in *ACM Workshop on Performance Metrics for Intelligent Systems*, 2008, pp. 57–61.

[29] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *IEEE International Conference on Computer Vision*, 1999, pp. 1033–1038.

[30] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[31] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations*, 2015.

[32] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[33] U. Pattacini, "Modular cartesian controllers for humanoid robots: Design and implementation on the iCub," PhD thesis, Istituto Italiano di Tecnologia, 2011.

[34] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[35] M. P. Deisenroth, P. Englert, J. Peters, and D. Fox, "Multi-task policy search for robotics," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 3876–3881.

[36] P. Peng, Y. Tian, T. Xiang, *et al.*, "Joint Semantic and Latent Attribute Modelling for Cross-Class Transfer Learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 7, pp. 1625–1638, 2018.